AMENDMENTS TO THE CLAIMS

Please cancel claim 1, without prejudice or disclaimer of the subject matter; add new claim 31; and amend claims 4, 10 to 15, 18, 19, 21, and 22, as shown below. This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

1. to 3. (Cancelled)

4. (Currently Amended) The frame handler of ~~claim 1~~ claim 31 wherein the data structure further comprises a tree structure .

5. (Previously Presented) The frame handler of claim 4 wherein the tree structure is an AVL tree.

6. (Previously Presented) The frame handler of claim 4 wherein the tree structure includes a frame node associated with each of the plurality of frames.

7. (Previously Presented) The frame handler of claim 6 wherein each frame node is associated with a list of unused instances within the associated frame.

8. (Previously Presented) The frame handler of claim 7 wherein the list of unused instances is represented as a ring structure.

9. (Previously Presented) The frame handler of claim 6 further comprising an anchor including:

an empty list storing each frame node having no unused instances; and

a non-empty list storing each frame node having unused instances.

10. (Currently Amended) The ~~frame handler of claim 1~~ system of claim 31 further comprising an operating system interface operable to allocate a block of memory such that the frame handler is operable to allocate an additional block of memory when the block of memory is exhausted.

11. (Currently Amended) A method for allocating memory in a computer system, the method comprising:

determining a size of a memory page used by a paged virtual memory system;

outputting a request from an application to ~~an operating system~~ the paged virtual memory for allocation of a block of memory by ~~the operating~~ an operating system to the application, the block of memory being integer *N* times the size of the memory page;

accessing the block of memory for the application;

dividing the block of memory into ~~a plurality of~~ (*N*-1) frames, with each of the ~~plurality of~~ frames operable to store an indexing structure associated with a single attribute of a data record, and each of the frames being the same size as the memory page used by the operating system;

determining a beginning page boundary of a first whole memory page within the block of memory;

storing the frames beginning at the beginning page boundary;

dividing each of the ~~plurality of~~ frames into a plurality of instances, with each of the plurality of instances operable to store an index node of the indexing structure, the index node including left and right pointers pointing to other index nodes of the index structure having the single attribute;

storing administrative data in a cut-off portion of the block of memory disposed in front of the beginning page boundary or behind the (N-1)*th* frame; and

maintaining a data structure identifying the unused instances within each of the ~~plurality of~~ frames.

12. (Currently Amended) The method of claim 11 wherein maintaining ~~a data structure~~ the data structure identifying the unused instances within each of the ~~plurality of~~ frames further comprises creating a frame node corresponding to each of the frames.

13. (Currently Amended) The method of claim 12 wherein maintaining ~~a data structure~~ the data structure identifying the unused instances within each of the ~~plurality of~~ frames further comprises associating a list of unused instances with each frame node.

14. (Currently Amended) The method of claim 13 wherein associating ~~a list~~ the list of unused instances with each frame node includes creating a ring data structure comprised of unused instances.

15. (Currently Amended) The method of claim 12 wherein maintaining ~~a data structure~~ the data structure identifying the unused instances further comprises organizing the frame nodes in a tree structure.

16. (Original) The method of claim 15 wherein the tree structure is an AVL tree.

17. (Original) The method of claim 12 further comprising creating an anchor data structure including a ring including an empty list and a non-empty list.

18. (Currently Amended) The method of claim 17 wherein maintaining ~~a data structure~~ the data structure identifying the unused instances further comprises placing frame nodes with unused instances in the non-empty list and placing nodes without unused instances in the empty list.

19. (Currently Amended) The method of claim 12 wherein dividing the block of memory into the ~~plurality of~~ (*N*-1) frames includes associating a frame identifier with each of the ~~plurality of~~ frames.

20. (Previously Presented) The method of claim 19 wherein each frame node includes the frame identifier of its associated frame.

21. (Currently Amended) A method comprising:

determining a size of a memory page used by a paged virtual memory system;

outputting a request from an application to the paged virtual memory for allocation of a block of memory by an operating system to the application, the block of memory being integer *N* times the size of the memory page;

accessing ~~a block~~ the block of memory for ~~an application~~ the application;

dividing the block of memory into ~~a plurality of~~ (*N*-1) frames, including first and second frames, with each of the ~~plurality of~~ frames operable to store an indexing structure associated with a single attribute of a data record, and each of the frames being the same size as the memory page used by the operating system;

determining a beginning page boundary of a first whole memory page within the block of memory;

storing each of the frames beginning at the beginning page boundary;

dividing each of the ~~plurality of~~ frames into a plurality of instances, including first and second lists of instances, with each of the plurality of instances operable to store an index node of the indexing structure, the index node including left and right pointers pointing to other index nodes of the index structure having the single attribute;

assigning a first identifier that is associated with the first frame to a first frame node;

linking the first list of instances to the first frame node;

assigning a second identifier that is associated with the second frame to a second frame node;

linking the second list of instances to the second frame node;

constructing a data structure using a plurality of nodes including the first node and the second node; ~~node; and~~

selecting available instances within each of the ~~plurality of~~ frames using the data structure, via ~~an application~~ the application; and

storing administrative data in a cut-off portion of the block of memory disposed in front of the beginning page boundary or behind the (N-1)*th* frame.

22. (Currently Amended) The method of claim 21 wherein constructing ~~a data structure~~ the data structure further comprises constructing an AVL tree using the plurality of frame nodes.

23. (Previously Presented) The method of claim 22 wherein selecting available instances using the data structure further comprises traversing the data structure to locate the available instances.

24. (Original) The method of claim 22 further comprising superposing a linear list over the data structure, wherein the linear list includes a first pointer to an empty subset of the plurality of nodes that has no associated memory available for use by the application and a second pointer to a not_empty subset that has associated memory available for use by the application.

25. (Previously Presented) The method of claim 24 wherein the first frame node is a first not_empty frame node in the not_empty subset, and wherein selecting available instances further comprises:

following the second pointer to the first frame node; and

using the first list of instances as the available instances.

26. (Previously Presented) The method of claim 25 further comprising:

re-setting the second pointer to a second not_empty frame node in the not_empty subset, and

including the first node in the empty subset.

27. (Original) The method of claim 21 further comprising:

determining an origin list from which the available instances were selected; and

returning the available instances to the origin list.

28. (Original) The method of claim 27 wherein determining the origin list comprises matching an identifier of the available instances to the first identifier or the second identifier.

29. (Previously Presented) The method of claim 28 wherein matching the identifier comprises following a pointer to a first not_empty frame node of a not_empty subset of the plurality of nodes, the not_empty subset including not_empty frame nodes with associated memory available for use by the application.

30. (Original) The method of claim 21 wherein the first memory portion includes a frame into which a block of memory allocated from the operating system is divided.

31. (New) A frame handler comprising:

an application configured to:

determine a size of a memory page used by a paged virtual memory system,

output a request from an application to the paged virtual memory for allocation of a block of memory by an operating system to the application, the block of memory being integer *N* times the size of the memory page,

access the block of memory for the application,

divide the block of memory into (*N*-1) frames, with each of the frames operable to store an indexing structure associated with a single attribute of a data record, and each of the frames being the same size as the memory page used by the operating system,

determining a beginning page boundary of a first whole memory page within the block of memory,

store each of the frames beginning at the beginning page boundary,

divide each of the frames into a plurality of instances, with each of the plurality of instances operable to store an index node of the indexing structure, the index node including left and right pointers pointing to other index nodes of the index structure having the single attribute,

store administrative data in a cut-off portion of the block of memory disposed in front of the beginning page boundary or behind the (N-1)*th* frame, and

maintain a data structure identifying the unused instances within each of the frames.